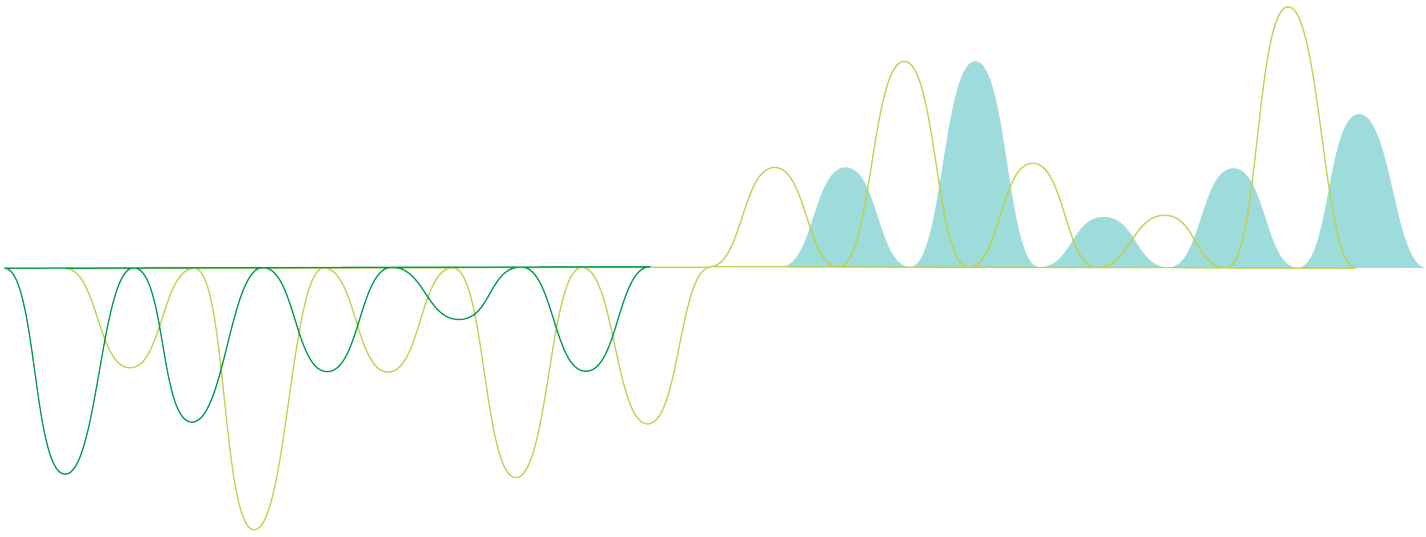
Учебное пособие — следующие этапы в построении скрипта

Qlik Sense®
May 2023



Contents

1 Учебное пособие. Добро пожаловать!

Рады вас приветствовать в нашем учебном пособии, которое познакомит вас с расширенными принципами написания скриптов в программе Qlik Sense.

Как только вы познакомитесь с основами написания скриптов, вы сможете начать выполнять более сложные операции с вашими данными при их загрузке в программу Qlik Sense. Это может включать, например, преобразование данных с использованием перекрестных таблиц, очистку данных, а также создание и загрузку данных из файлов данных Qlik, известных как файлы QVD.

1.1 Что вы узнаете

After completing this tutorial, you should be comfortable with loading data using some of the more advanced scripting functions in Qlik Sense.

1.2 Кому следует пройти этот курс

Необходимо знать основы написания скриптов в Qlik Sense. То есть необходимо иметь опыт загрузки данных и управления ими с использованием скриптов.

При отсутствии такого опыта рекомендуется изучить учебное пособие «Написание скриптов для начинающих».

Требуется доступ к редактору загрузки данных, а также разрешение на загрузку данных в Qlik Sense Enterprise on Windows.

Инструкции также применимы в общем для Qlik Sense Cloud Business.

1.3 Содержимое пакета

В загруженном пакете zip содержатся следующие файлы данных, которые вам потребуются для изучения учебного пособия:

- *Cutlery.xlsx*
- *Data.xlsx*
- *Events.txt*
- *Employees.xlsx*
- *Intervals.txt*
- *Product.xlsx*
- *Salesman.xlsx*
- *Transactions.csv*
- *Winedistricts.txt*

В пакете также содержится копия приложения *Учебное пособие по написанию скриптов продвинутого уровня*. Дополнительные разделы скриптов в приложении содержат скрипты для других приложений, которые вы создаете при изучении этого учебного пособия. Вы можете загрузить приложение в свой хаб.

Рекомендуется разрабатывать приложение самостоятельно, как описано в учебном пособии, для более эффективного обучения. Кроме того, для работы вам нужно будет загрузить файлы данных и подключиться к ним, как описано в руководстве по загрузке данных.

Но если у вас возникнут проблемы, приложение может помочь вам их решить. Мы указали, с каким уроком связаны какие сегменты скрипта.

1.4 Уроки, содержащиеся в учебном пособии

В зависимости от опыта работы с Qlik Sense для изучения этого учебного пособия потребуется 3–4 часа. Темы должны изучаться последовательно. Однако вы всегда можете прерваться и в любое время снова вернуться к ним. К счастью, тестов здесь нет.

Преобразование данных

Использование префикса Crosstable

Объединение таблиц с помощью операторов Join и Keep

Использование функций между записями: Peek, Previous и Exists

Интервалы сопоставления и итеративная загрузка

Очистка данных

Обработка иерархических данных

Файлы QVD

1.5 Дополнительные информационные ресурсы

- Программа Qlik предлагает широкий набор ресурсов для дополнительного изучения.
- Доступна интерактивная справка Qlik.
- Обучение, в том числе бесплатные интерактивные курсы, доступно в разделе Qlik Continuous Classroom.
- Дискуссионные форумы, блоги и многое другое находится в разделе Qlik Community.

2 Операторы LOAD и SELECT

Данные можно загружать в программу Qlik Sense с помощью операторов LOAD и SELECT. Каждый из них создает внутреннюю таблицу. Оператор LOAD используется для загрузки данных из файлов, а оператор SELECT — для загрузки данных из баз данных.

В этом учебном пособии будут использоваться данные из файлов, поэтому будут применяться операторы LOAD.

Кроме того, можно использовать предыдущую загрузку LOAD для манипулирования содержимым загружаемых данных. Например, переименование полей должно выполняться в операторе LOAD, в то время как оператор SELECT не допускает каких-либо изменений в именах полей.

При загрузке данных в программу Qlik Sense применяются следующие правила:

- Программа Qlik Sense не делает различия между таблицами, созданными операторами LOAD и SELECT. Таким образом, если загружается несколько таблиц, то не имеет значения, загружены они с помощью оператора LOAD или SELECT либо комбинации этих двух операторов.
- Порядок полей в операторе или исходной таблице базы данных не имеет значения для логики программы Qlik Sense.
- Имена полей зависят от регистра и используются для установления связей между таблицами данных. В связи с этим иногда необходимо переименовывать поля в скрипте загрузки для получения желаемой модели данных.

3 Преобразование данных

Перед использованием данных в приложении можно преобразовывать данные и управлять ими в редакторе загрузки данных.

Одним из преимуществ управления данными является возможность выбрать только загрузку подмножества данных из файла, например нескольких выбранных столбцов из таблицы, чтобы обработка данных выполнялась более эффективно. Данные можно загрузить несколько раз, чтобы разбить необработанные данные на несколько новых логических таблиц. Также можно загрузить данные из нескольких источников и объединить их в одной таблице в программе Qlik Sense.

В следующих упражнениях показано, как загружать данные с использованием префикса Crosstable. Также вы узнаете, как объединять таблицы, использовать функции между записями, такие как Peek и Previous, и загружать одну и ту же строку несколько раз с помощью оператора While Load.

3.1 Использование префикса Crosstable

Перекрестные таблицы — распространенный тип таблиц, включающих матрицу значений, расположенную между двумя ортогональными списками данных в заголовках. В данных перекрестной таблицы префикс Crosstable можно использовать для преобразования данных и создания желаемых полей.

Префикс Crosstable

В следующей таблице *Product* есть по одному столбцу для каждого месяца и по одной строке для каждого продукта.

Таблица продуктов

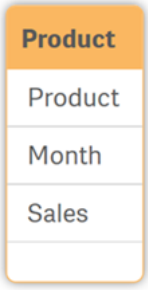
Продукт	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

После загрузки этой таблицы в результате будет получена таблица с одним полем для элемента *Product* и по одному полю для каждого месяца.

Таблица Product, содержащая поле Product и по одному полю для каждого месяца

При необходимости проанализировать эти данные удобнее иметь все числа в одном поле, а все месяцы в другом. В таком случае это таблица из трех столбцов, с одним столбцом для каждой категории (*Product, Month, Sales*).

Таблица Product с полями Product, Month и Sales



Префикс Crosstable преобразует данные в таблицу с одним столбцом для категории *Month* и другим столбцом для категории *Sales*. Иначе говоря, имена полей преобразовываются в значения полей.

Выполните следующие действия.

1. Создайте новое приложение и назовите его *Учебное пособие по написанию скриптов продвинутого уровня*.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Название раздела *Product*.
4. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
5. Загрузите, а затем выберите *Product.xlsx*.
6. Выберите таблицу *Product* в окне **Выбрать данные из**.

   *Убедитесь, что в разделе **Имена полей** выбран параметр **Встроенные имена полей**, чтобы включить имена полей таблицы при загрузке данных.*

7. Щелкните команду **Вставить скрипт**.

Скрипт должен выглядеть следующим образом:

```
LOAD     Product,     "Jan 2014",     "Feb 2014",     "Mar 2014",     "Apr 2014",
"May 2014",     "Jun 2014" FROM [lib://AttachedFiles/Product.xlsx]  (ooxml, embedded
labels, table is Product);
```

8. Щелкните команду **Загрузить данные**.
9. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:

   Таблица Product, содержащая поле Product и по одному полю для каждого месяца

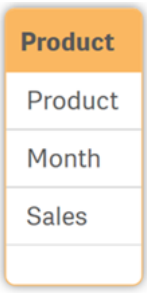
   

10. Щелкните вкладку *Product* в **редакторе загрузки данных**.
11. Введите следующее над оператором LOAD:

    ```
    CrossTable(Month, Sales)
    ```

12. Щелкните команду **Загрузить данные**.
13. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:

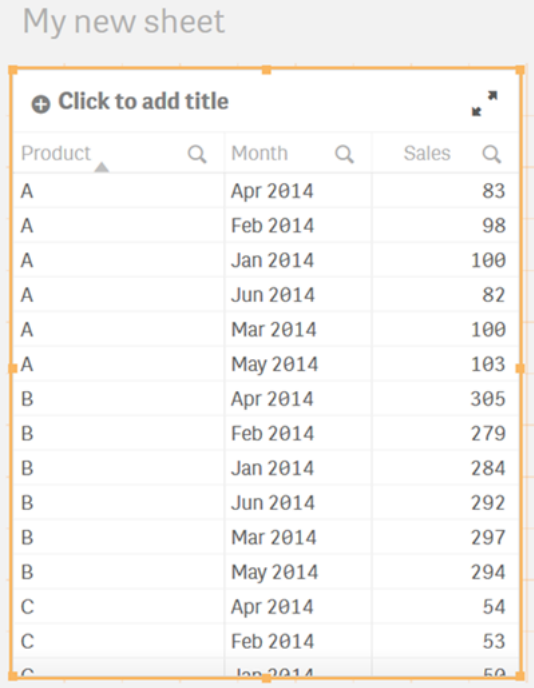
    Таблица Product с полями Product, Month и Sales

Обратите внимание, что обычно входные данные имеют только один столбец в качестве поля-квалификатора; в качестве внутреннего ключа (*Product* в примере выше). Но можно иметь несколько. Если так, то все поля-квалификаторы должны быть указаны перед полями атрибутов в операторе LOAD, а третий параметр для префикса Crosstable необходимо использовать для определения числа полей-квалификаторов. Нельзя ставить предшествующий оператор LOAD или префикс перед ключевым словом Crosstable. Однако можно использовать автоматическое объединение.

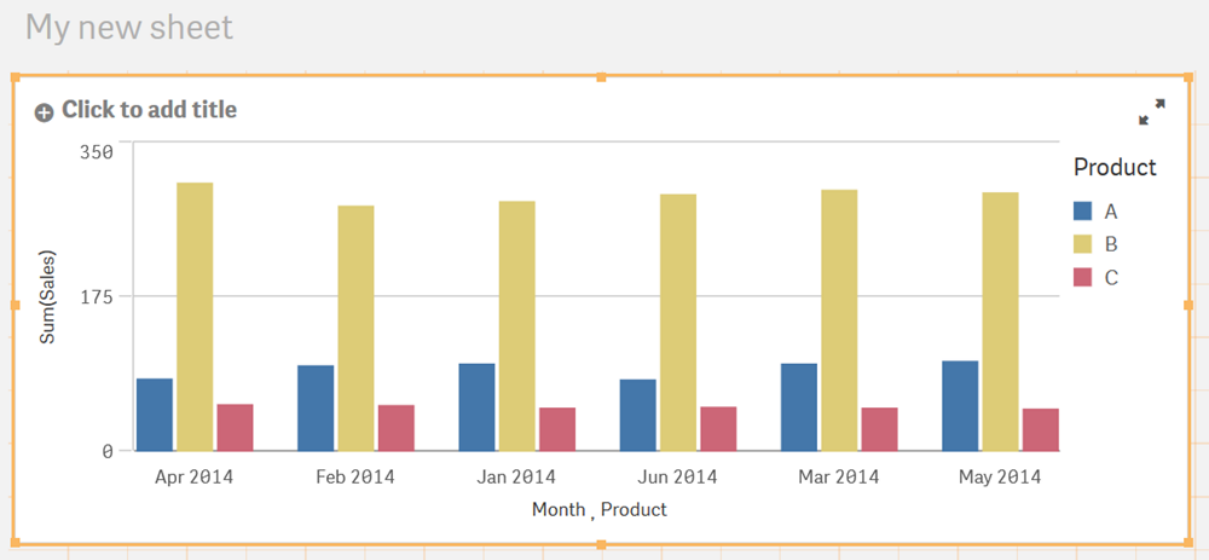
Данные таблицы в Qlik Sense выглядят следующим образом:

Таблица, в которой отображаются данные, загруженные с помощью префикса Crosstable



Теперь вы можете, например, создать линейчатую диаграмму, используя следующие данные:

Линейчатая диаграмма, отображающая данные, загруженные с использованием префикса Crosstable



Чтобы узнать дополнительные сведения о функции Crosstable, см. эту запись блога в Qlik Community: The Crosstable Load (Загрузка с использованием префикса Crosstable). Поведение обсуждается в контексте QlikView. Однако логика в равной степени относится к Qlik Sense.

Числовая интерпретация не работает для полей атрибутов. Это значит, что если в качестве заголовков столбцов указаны месяцы, они не будут автоматически интерпретированы. Обходной прием с целью использования префикса Crosstable для создания временной таблицы и запуска второго прохода по ней, чтобы выполнить интерпретации, как показано в следующем примере.

Обратите внимание, что это просто пример. Здесь нет сопутствующих упражнений, которые следует выполнить в Qlik Sense.

```
tmpData: Crosstable (MonthText, Sales) LOAD Product, [Jan 2014], [Feb 2014], [Mar 2014], [Apr
2014], [May 2014], [Jun 2014]            FROM ...    Final: LOAD Product, Date(Date#(MonthText,'MMM
YYYY'),'MMM YYYY') as Month, Sales Resident tmpData; Drop Table tmpData;
```

Очистка кэша памяти

Чтобы очистить кэш памяти, созданные таблицы можно удалить. При использовании временной таблицы для выполнения загрузки, как в предыдущем разделе, эту таблицу можно удалить, когда в ней больше не будет необходимости. Пример.

```
DROP TABLE Table1, Table2, Table3, Table4; DROP TABLES Table1, Table2, Table3, Table4;
```

Можно также удалить поля. Пример.

```
DROP FIELD Field1, Field2, Field3, Field4; DROP FIELDS Field1, Field2, Field3, Field4; DROP
FIELD Field1 from Table1; DROP FIELDS Field1 from Table1;
```

Как видите, ключевые слова TABLE и FIELD могут иметь как единственное, так и множественное число.

3.2 Объединение таблиц с помощью операторов Join и Keep

Объединение — операция объединения двух таблиц в одну. Записи результирующей таблицы представляют собой комбинации записей в исходных таблицах. При этом две такие записи, составляющие одну комбинацию в результирующей таблице, как правило, имеют общее значение одного или нескольких общих полей. Такое объединение называется естественным. В программе Qlik Sense объединение может выполняться в скрипте, создавая логическую таблицу.

Таблицы, которые находятся в скрипте, можно объединять. Логика Qlik Sense будет распознавать не отдельные таблицы, а результаты объединения, которые будут представлены в одной внутренней таблице. В некоторых случаях это требуется, однако существуют недостатки:

- Загруженные таблицы часто становятся больше, и программа Qlik Sense работает медленнее.
- Некоторая информация может быть потеряна: частота (количество записей) в исходной таблице может быть больше недоступна.

Функция Keep, которая позволяет уменьшить одну или обе таблицы до пересечения данных таблиц перед сохранением таблиц в программу Qlik Sense, предназначена для уменьшения количества случаев, когда необходимо использовать явные объединения.

В данном руководстве термин «объединение» обычно используется для объединений, выполненных до создания внутренних таблиц. Однако ассоциация, выполненная после создания внутренних таблиц, по сути, также является объединением.

Join

Самым простым способом создания объединения является использование префикса Join в скрипте, который позволяет объединять внутреннюю таблицу с другой именованной таблицей или последней созданной таблицей. Объединение будет внешним и позволит создать все возможные сочетания значений из двух таблиц.

Пример:

```
LOAD  a, b, c from table1.csv; join LOAD a, d from table2.csv;
```

Результирующая внутренняя таблица имеет поля a, b, c и d. Количество записей различается в зависимости от значений полей этих двух таблиц.

Имена объединяемых полей должны совпадать. Количество объединяемых полей может быть любым. Обычно в таблицах должно быть одно или несколько общих полей. При отсутствии общих полей будет рассматриваться декартово произведение таблиц. В принципе все поля могут быть общими, однако обычно в этом нет смысла. Пока имя ранее загруженной таблицы не будет указано в операторе Join, префиксом Join будет использоваться последняя созданная таблица. Поэтому порядок двух операторов не является произвольным.

Использование Join

Явный префикс Join в языке скриптов в программе Qlik Sense выполняет полное объединение двух таблиц. В результате получается одна таблица. Выполнение таких объединений часто может приводить к получению очень больших таблиц.

Выполните следующие действия.

1. Откройте приложение *Учебное пособие по написанию скриптов продвинутого уровня*.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите раздел *Transactions*.
4. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
5. Загрузите, а затем выберите *Transactions.csv*.

   *Убедитесь, что в разделе **Имена полей** выбран параметр **Встроенные имена полей**, чтобы включить имена полей таблицы при загрузке данных.*

6. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.

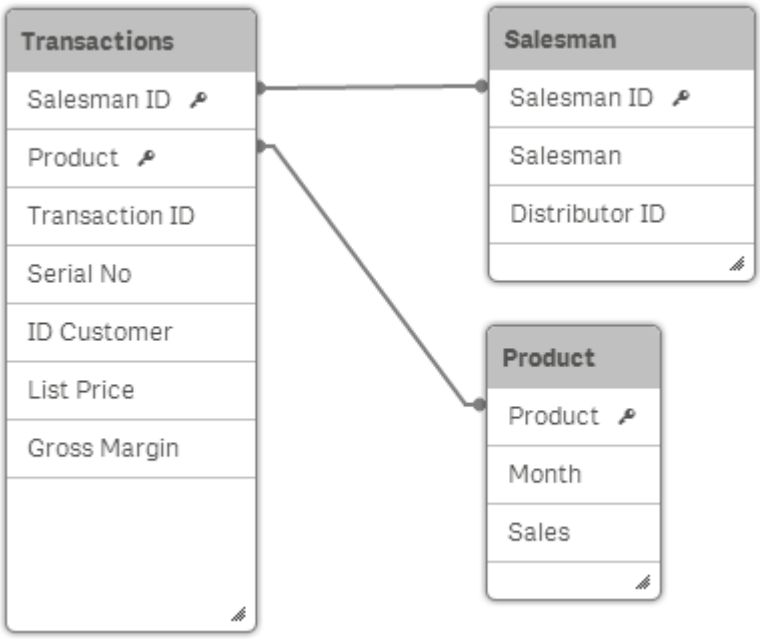
7. Загрузите, а затем выберите *Salesman.xlsx*.
8. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.

   Скрипт должен выглядеть следующим образом:

   ```
   LOAD     "Transaction ID",     "Salesman ID",     Product,     "Serial No",     "ID
   Customer",     "List Price",     "Gross Margin" FROM
   [lib://AttachedFiles/Transactions.csv] (txt, codepage is 28591, embedded labels,
   delimiter is ',', msq);  LOAD     "Salesman ID",     Salesman,     "Distributor ID" FROM
   [lib://AttachedFiles/Salesman.xlsx] (ooxml, embedded labels, table is Salesman);
   ```

9. Щелкните команду **Загрузить данные**.
10. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:

    Модель данных: Таблицы Transactions, Salesman и Product



Однако результат, полученный в результате разделения таблиц *Transactions* и *Salesman*, может быть нежелательным. Возможно, лучше объединить две таблицы.

Выполните следующие действия.

1. Чтобы задать имя объединенной таблицы, добавьте следующую строку над первым оператором LOAD:

   ```
   Transactions:
   ```

2. Чтобы объединить таблицы *Transactions* и *Salesman*, добавьте следующую строку над вторым оператором LOAD:

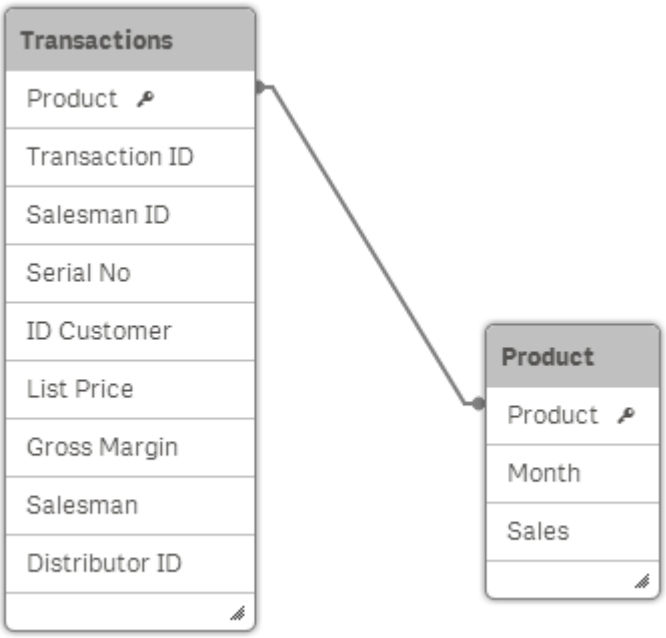
   ```
   Join(Transactions)
   ```

   Скрипт должен выглядеть следующим образом:

```
Transactions: LOAD     "Transaction ID",     "Salesman ID",     Product,     "Serial
No",     "ID Customer",     "List Price",     "Gross Margin" FROM
[lib://AttachedFiles/Transactions.csv] (txt, codepage is 28591, embedded labels,
delimiter is ',', msq);  Join(Transactions) LOAD     "Salesman ID",     Salesman,
"Distributor ID" FROM [lib://AttachedFiles/Salesman.xlsx] (ooxml, embedded labels, table
is Salesman);
```

3. Щелкните команду **Загрузить данные**.
4. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:

   Модель данных: Таблицы Transactions и Product



   Все поля таблиц *Transactions* и *Salesman* теперь будут объединены в одну таблицу *Transactions*.

> *Чтобы узнать о том, когда можно использовать префикс Join, см. эти записи блога в сообществе Qlik Community:* Объединять или не объединять, Сопоставление как альтернатива объединению. *Поведение обсуждается в контексте QlikView. Однако логика в равной степени относится к Qlik Sense.*

Keep

Одной из основных функций программы Qlik Sense является способность к связыванию таблиц вместо их объединения, что позволяет сократить использование памяти, повысить скорость обработки и гибкость. Функция Keep предназначена для сокращения числа случаев необходимого использования явных объединений.

Префикс Keep между двумя операторами LOAD или SELECT приводит к уменьшению одной или обеих таблиц до пересечения их данных перед сохранением таблиц в программе Qlik Sense. Перед префиксом

Keep следует задать одно из ключевых слов: Inner, Left или Right. Выборка записей из таблицы осуществляется так же, как и при соответствующем объединении. Однако две таблицы не объединяются и сохраняются в программе Qlik Sense в виде двух отдельных именованных таблиц.

Inner

Перед префиксами Join и Keep в скрипте загрузки данных можно использовать префикс Inner.

При использовании этого префикса перед префиксом Join объединение двух таблиц будет внутренним. Полученная таблица содержит только сочетания из двух таблиц, включающие полный набор данных с обеих сторон.

Если этот префикс используется перед Keep, он указывает, что две таблицы следует уменьшить до области взаимного пересечения, прежде чем они смогут быть сохранены в программе Qlik Sense.

Пример:

В этих примерах используются исходные таблицы *Table1* и *Table2*.

Обратите внимание, что это просто примеры. Здесь нет сопутствующих упражнений, которые следует выполнить в Qlik Sense.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

Сначала выполняется Inner Join в отношении таблиц, в результате чего образуется таблица *VTable*, содержащая только одну строку, только одну запись, существующую в обеих таблицах, с данными из обеих таблиц.

```
VTable: SELECT * from Table1; inner join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx

Inner Keep

Если вместо этого выполнить Inner Keep, таблиц все равно будет две. Две таблицы связаны посредством общего поля *A*.

```
VTab1: SELECT * from Table1; VTab2: inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

Перед префиксами Join и Keep в скрипте загрузки данных можно использовать префикс left.

При использовании этого префикса перед префиксом Join объединение двух таблиц будет левосторонним. Полученная таблица содержит только сочетания из двух таблиц, включающие полный набор данных из первой таблицы.

Если этот префикс используется перед префиксом Keep, он указывает, что вторую таблицу следует уменьшить до области взаимного пересечения с первой таблицей перед сохранением в программе Qlik Sense.

Пример:

В этих примерах используются исходные таблицы *Table1* и *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Сначала выполняется Left Join в отношении таблиц, в результате чего образуется таблица *VTable*, содержащая все строки из таблицы *Table1*, совмещенные с полями из совпадающих строк в таблице *Table2*.

```
VTable: SELECT * from Table1; left join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
2	cc	-
3	ee	-

Если вместо этого выполнить Left Keep, таблиц все равно будет две. Две таблицы связаны посредством общего поля *A*.

```
VTab1: SELECT * from Table1; VTab2: left keep SELECT * from Table2;
```

VTab1

A	B
1	aa
2	cc
3	ee

VTab2

A	C
1	xx

Right

Перед префиксами Join и Keep в языке скриптов Qlik Sense можно использовать префикс right.

При использовании этого префикса перед префиксом Join объединение двух таблиц будет правосторонним. Полученная таблица содержит только сочетания из двух таблиц, включающие полный набор данных из второй таблицы.

Если этот префикс используется перед префиксом Keep, он указывает, что первую таблицу следует уменьшить до области взаимного пересечения со второй таблицей перед сохранением в программе Qlik Sense.

Пример:

В этих примерах используются исходные таблицы *Table1* и *Table2*.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Сначала выполняется Right Join в отношении таблиц, в результате чего образуется таблица *VTable*, содержащая все строки из таблицы *Table2*, совмещенные с полями из совпадающих строк в таблице *Table1*.

```
VTable: SELECT * from Table1; right join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
4	-	yy

Если вместо этого выполнить Right Keep, таблиц все равно будет две. Две таблицы связаны посредством общего поля *A*.

```
VTab1: SELECT * from Table1; VTab2: right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 Использование функций между записями: Peek, Previous и Exists

Эти функции используются, если для оценки текущей записи требуется значение из ранее загруженных записей данных.

В этой части учебного пособия мы изучим функции Peek(), Previous() и Exists().

Peek()

Функция **Peek()** возвращает значение поля в таблице для строки, которая уже загружена. Можно указать номер строки или таблицу. Если номер строки не указан, будет использована последняя запись, загруженная ранее.

Синтаксис:
```
Peek(fieldname [ , row [ , tablename ] ] )
```

Элемент Row должен быть целым числом. 0 обозначает первую запись, 1 обозначает вторую и т. д. Отрицательные числа указывают порядок с конца таблицы. -1 обозначает последнюю прочитанную запись.

Если элемент row не указан, принимается значение -1.

Tablename является меткой таблицы без двоеточия на конце. Если элемент *tablename* не указан, принимается текущая таблица. При использовании вне оператора **LOAD** или в ссылке на другую таблицу должен включаться элемент *tablename*.

Previous()

Функция **Previous()** находит значение выражения **expr** с помощью данных из ранее введенной записи, которая не была сброшена из-за предложения **where**. В первой записи внутренней таблицы функция возвратит значение NULL.

Синтаксис:
```
Previous(expression)
```

Функцию Previous() можно использовать вложенным образом, чтобы получить доступ к более ранним записям. Данные выбираются из входного источника напрямую, что также позволяет ссылаться на поля, которые не были загружены в программу Qlik Sense, то есть даже если они не были сохранены в связанной базе данных.

Exists()

Функция **Exists()** определяет, загружено ли определенное значение поля в поле в скрипте загрузки данных. Функция возвращает значение TRUE или FALSE, таким образом, ее можно использовать в предложении **where** оператора **LOAD** или **IF**.

Синтаксис:
```
Exists(field [, expression ] )
```

Поле должно существовать в данных, загруженных скриптом к текущему времени. *Expression* — выражение, которое вычисляет значение поля для поиска в указанном поле. При его отсутствии принимается значение текущей записи в указанном поле.

Использование функций Peek() и Previous()

В своем простом виде функции Peek() и Previous() используются для указания определенных значений в таблице. В таблице *Employees* представлена выборка данных, которые будут загружены в этом упражнении.

Выборка данных из таблицы Employees

Дата	**Hired**	**Terminated**
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

На данный момент происходит сбор данных только для месяцев, найма и увольнения, поэтому мы собираемся добавить поля для элементов *Employee Count* и *Employee Var* с помощью функций Peek() и Previous(), чтобы увидеть месячную разницу по сотрудникам в целом.

Выполните следующие действия.

1. Откройте приложение *Учебное пособие по написанию скриптов продвинутого уровня*.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите раздел *Employees*.
4. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
5. Загрузите, а затем выберите *Employees.xlsx*.

   Убедитесь, что в разделе Field names выбран параметр Embedded field names, чтобы включить имена полей таблицы при загрузке данных.

6. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.

   Скрипт должен выглядеть следующим образом:

   ```
   LOAD        "Date",     Hired,     Terminated FROM [lib://AttachedFiles/Employees.xlsx]
   (ooxml, embedded labels, table is Sheet1);
   ```

7. Измените скрипт, чтобы он выглядел так:
   ```
   [Employees Init]: LOAD     rowno() as Row,     Date(Date) as Date,     Hired,
   Terminated,     If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-
   Terminated)) as [Employee Count] FROM [lib://AttachedFiles/Employees.xlsx]
   embedded labels, table is Sheet1);
   ```

Даты в поле данных *Date* на листе Excel имеют формат ММ/ДД/ГГГГ. Чтобы обеспечить правильное распознавание дат с помощью формата, указанного в системных переменных, к полю Date необходимо применить функцию *Date*.

Функция Peek() позволяет находить любое значение, загруженное для указанного поля. В первую очередь следует проверить, равняется ли значение rowno() единице (1). Если оно равняется 1, то элемента *Employee Count* не будет, поэтому поле заполняется значением разницы между *Hired* и *Terminated*.

Если значение rowno() больше 1, мы посмотрим на элемент *Employee Count* за последний месяц и используем это число, чтобы добавить его к значению разницы элемента *Hired* за этот месяц минус количество сотрудников из элемента *Terminated*.

Также обратите внимание, что в функции Peek() мы используем (-1). Поэтому программа Qlik Sense должна обратиться к записи над текущей записью. Если значение (-1) не указано, программа Qlik Sense будет считать, что вы хотите увидеть предыдущую запись.

8. Добавьте следующее выражение в конец скрипта:

```
[Employee Count]: LOAD Row, Date, Hired, Terminated, [Employee Count], If(rowno
()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var] Resident
[Employees Init] Order By Row asc; Drop Table [Employees Init];
```

Функция Previous() позволяет находить последнее значение, загруженное для указанного поля. В первую очередь следует проверить, равняется ли значение rowno() единице (1). Если оно равняется 1, то элемента Employee Var не будет, поскольку для элемента *Employee Count* предыдущего месяца нет записей. Поэтому мы просто вводим 0 для данного значения.

Если элемент rowno() больше 1, то будет элемент *Employee Var*, поэтому мы посмотрим на элемент *Employee Count* последнего месяца и вычтем это число из элемента *Employee Count* для текущего месяца, чтобы создать значение в поле *Employee Var*.

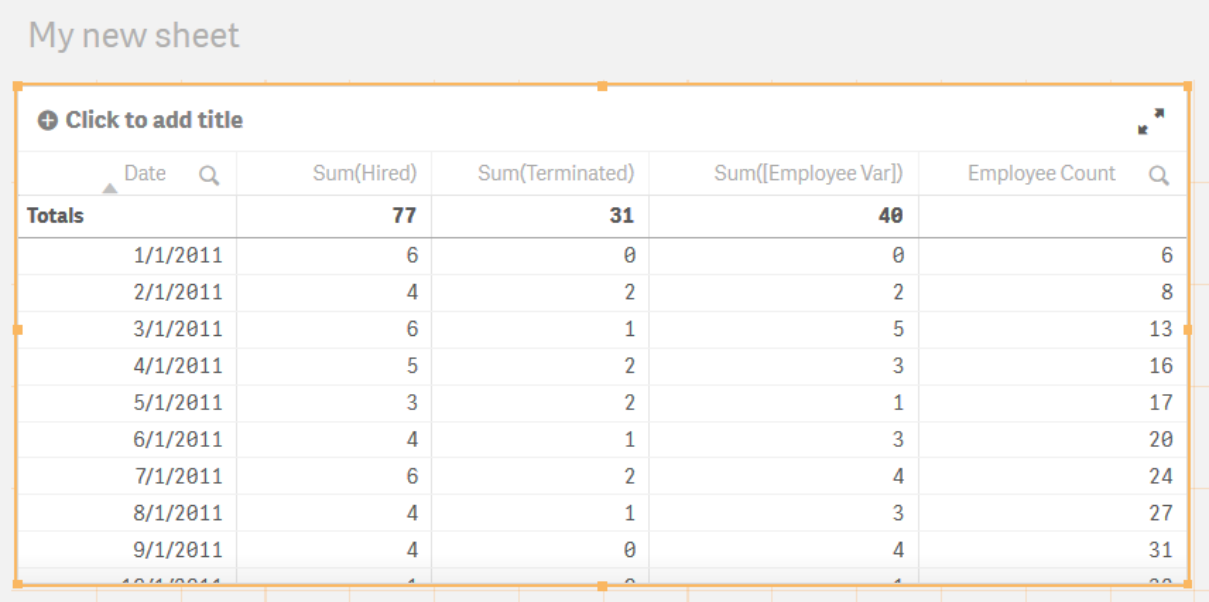
Скрипт должен выглядеть следующим образом:

```
[Employees Init]: LOAD rowno() as Row, Date(Date) as Date, Hired,
Terminated, If(rowno()=1, Hired-Terminated, peek([Employee Count], -1)+(Hired-
Terminated)) as [Employee Count] FROM [lib://AttachedFiles/Employees.xlsx] (ooxml,
embedded labels, table is Sheet1); [Employee Count]: LOAD Row, Date, Hired,
    Terminated, [Employee Count], If(rowno()=1,0,[Employee Count]-Previous
([Employee Count])) as [Employee Var] Resident [Employees Init] Order By Row asc; Drop
Table [Employees Init];
```

9. Щелкните команду **Загрузить данные**.

На новом листе обзора приложения создайте таблицу, используя поля *Date, Hired, Terminated, Employee Count* и *Employee Var* в качестве столбцов таблицы. Полученная таблица должна выглядеть следующим образом:

В следующей таблице в скрипте используются функции Peek и Previous



Date	Sum(Hired)	Sum(Terminated)	Sum([Employee Var])	Employee Count
Totals	**77**	**31**	**40**	
1/1/2011	6	0	0	6
2/1/2011	4	2	2	8
3/1/2011	6	1	5	13
4/1/2011	5	2	3	16
5/1/2011	3	2	1	17
6/1/2011	4	1	3	20
7/1/2011	6	2	4	24
8/1/2011	4	1	3	27
9/1/2011	4	0	4	31

Функции Peek() и Previous() позволяют намечать определенные строки в таблице. Самая большая разница между этими двумя функциями заключается в том, что функция Peek() позволяет пользователю увидеть поле, которое еще не загружено в скрипт, тогда как функция Previous() позволяет увидеть только уже загруженное поле. Функция Previous() работает с входными данными оператора LOAD, а функция Peek() работает с выходными данными оператора LOAD. (Такая же разница, как между элементами RecNo() и RowNo().) Это значит, что эти две функции будут вести себя по-разному, если есть предложение Where.

Поэтому функцию Previous() лучше использовать в ситуации, когда необходимо показать разницу текущего значения с предыдущим. В примере мы вычисляли, как изменялись данные по сотрудникам из месяца в месяц.

Функцию Peek() лучше использовать, когда целью является поле, которое еще не загружено в таблицу, или когда требуется наметить определенную строку. Это было показано в примере, где мы вычисляли элемент *Employee Count*, посмотрев данные элемента *Employee Count* предыдущего месяца, а затем добавили разницу между нанятыми и уволенными сотрудниками для текущего месяца. Помните, что в оригинальном файле не было поля *Employee Count*

Чтобы узнать дополнительные сведения об использовании функций Peek() и Previous(), см. эту запись блога в Qlik Community: Peek() vs Previous() – When to Use Each. *Поведение обсуждается в контексте QlikView. Однако логика в равной степени относится к Qlik Sense.*

Использование Exists()

Функция Exists() часто используется с предложением Where в скрипте, чтобы загружать данные, если соответствующие данные уже загружены в модель данных.

В следующем примере мы также используем функцию Dual(), чтобы назначить числовые значения строкам.

Выполните следующие действия.

1. Создайте новое приложение и дайте ему имя.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите раздел *People*.
4. Введите следующий скрипт:

```
//Add dummy people data PeopleTemp: LOAD * INLINE [ PersonID, Person 1, Jane 2, Joe 3, Shawn 4, Sue 5, Frank 6, Mike 7, Gloria 8, Mary 9, Steven, 10, Bill ];   //Add dummy age data AgeTemp: LOAD * INLINE [ PersonID, Age 1, 23 2, 45 3, 43 4, 30 5, 40 6, 32 7, 45 8, 54 9, 10, 61 11, 21 12, 39 ];   //LOAD new table with people People: NoConcatenate LOAD    PersonID,     Person Resident PeopleTemp;   Drop Table PeopleTemp;   //Add age and age bucket fields to the People table Left Join (People) LOAD     PersonID,     Age,    If (IsNull(Age) or Age='', Dual('No age', 5),       If(Age<25, Dual('Under 25', 1),          If (Age>=25 and Age <35, Dual('25-34', 2),            If(Age>=35 and Age<50, Dual('35-49' , 3),   If(Age>=50, Dual('50 or over', 4)          ))))) as AgeBucket Resident AgeTemp Where Exists(PersonID);   DROP Table AgeTemp;
```

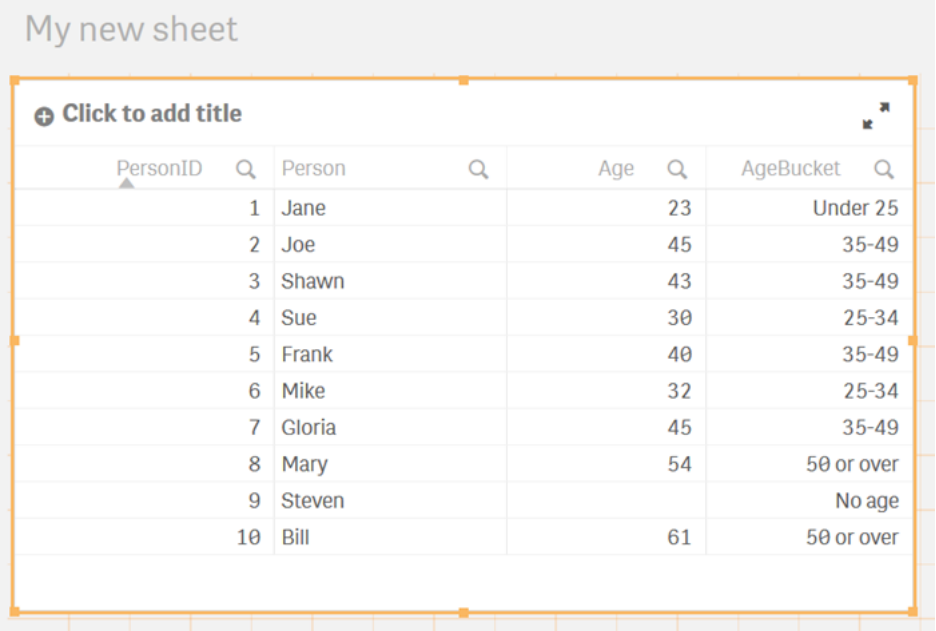
5. Щелкните команду **Загрузить данные**.

   В скрипте поля *Age* и *AgeBucket* загружаются, только если поле *PersonID* уже загружено в модель данных.

   Обратите внимание на таблицу *AgeTemp*. В ней для поля *PersonID* указаны значения возраста 11 и 12, но поскольку эти идентификаторы не были загружены в модель данных (в таблице *People*), они исключены предложением Where Exists(PersonID). Это предложение также можно записать следующим образом: Where Exists(PersonID, PersonID).

   Результат работы скрипта имеет следующий вид:

В следующей таблице в скрипте используется функция Exists

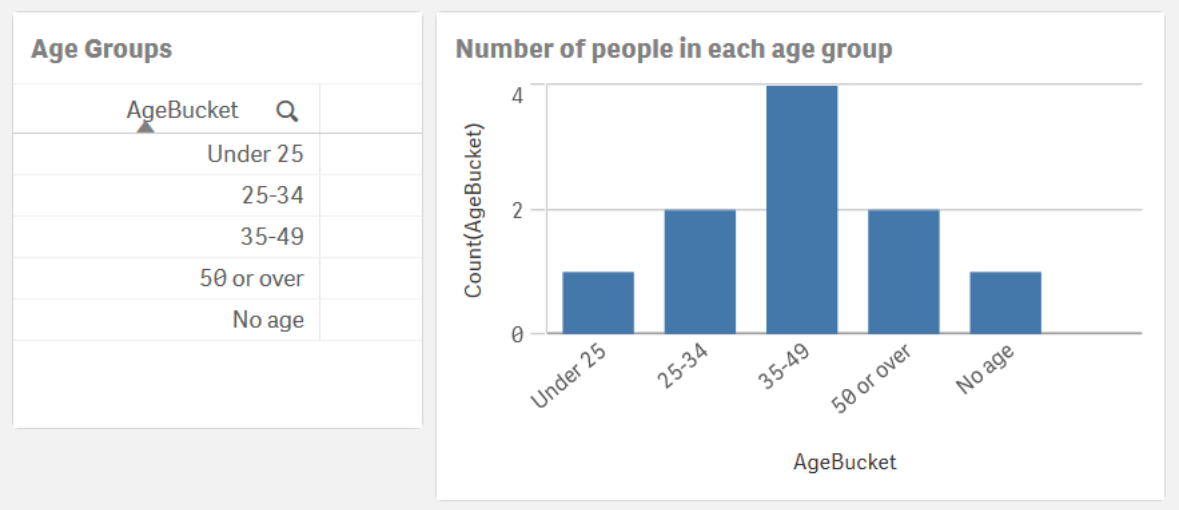


PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

Если бы ни одно значение поля *PersonID* в таблице *AgeTemp* не было загружено в модель данных, то поля *Age* и *AgeBucket* не были бы объединены в таблицу *People*. Использование функции Exists () может помочь предотвратить потерю записей/данных в модели данных, т. е. поля *Age* и *AgeBucket*, с которыми никто из людей не связан.

6. Создайте новый лист и присвойте ему имя.
7. Откройте новый лист и щелкните **Изменить лист**.
8. Добавьте стандартную таблицу на лист с измерением *AgeBucket* и назначьте визуализации имя *Возрастные группы.*
9. Добавьте линейчатую диаграмму на лист с измерением *AgeBucket* и мерой *Count([AgeBucket])*. Назовите визуализацию *Number of people in each age group*.
10. Измените свойства таблицы и линейчатой диаграммы согласно своим предпочтениям, а затем щелкните **Готово**.
    Полученный лист будет выглядеть примерно так:

Лист с группировками по возрасту



Функция Dual() очень полезна в скрипте или в выражении диаграммы, когда необходимо назначить строке числовое значение.

В скрипте выше есть приложение, которое загружает значения возраста. Вы решили поместить эти значения в блоки, поэтому можете создавать визуализации на основе противопоставления блоков со значениями возраста фактическим значениям возраста. Есть блок для людей младше 25 лет, от 25 до 35 лет и так далее. С помощью функции Dual() блокам со значениями возраста можно назначить числовое значение, которое позднее можно использовать для сортировки блоков со значениями возраста в списке или диаграмме. Поэтому, как на листе приложения, в результате сортировки в конце списка появляется сообщение «Нет возраста».

Чтобы узнать дополнительные сведения о функциях Exists() и Dual(), см. эту запись блога в Qlik Community: Dual & Exists – Useful Functions *(Dual и Exists — полезные функции)*

3.4 Интервалы сопоставления и итеративная загрузка

Префикс Intervalmatch для операторов LOAD или SELECT используется для связывания дискретных числовых значений с одним или несколькими числовыми интервалами. Это очень полезная функция, которая может использоваться, например, в производственных средах.

Использование префикса IntervalMatch()

Основной пример сопоставления интервалов, это когда в одной таблице перечислены числа или даты (события), а в другой — интервалы. Цель — связать две таблицы. В целом, это отношение между множествами, т. е. к интервалу может относиться много дат, а дата может относиться ко многим интервалам. Для решения этой проблемы необходимо создать таблицу пересчета двух исходных таблиц. Это можно сделать несколькими способами.

Самый простой способ для решения этой проблемы в программе Qlik Sense — использовать префикс IntervalMatch() перед оператором LOAD или SELECT. Оператор LOAD/SELECT должен содержать только два поля: From и To, которые используются для определения интервалов. Префикс IntervalMatch() создаст все возможные комбинации загруженных интервалов с ранее загруженными числовыми полями, указанными в качестве параметра для префикса.

Выполните следующие действия.

1. Создайте новое приложение и дайте ему имя.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите разделы *Events*.
4. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
5. Загрузите, а затем выберите *Events.txt*.
6. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.
7. Загрузите, а затем выберите *Intervals.txt*.
8. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.
9. В скрипте назовите первую таблицу *События*, а вторую таблицу — *Intervals*.
10. В конце скрипта добавьте оператор IntervalMatch, чтобы создать третью таблицу, которая выполнит пересчет двух первых таблиц:

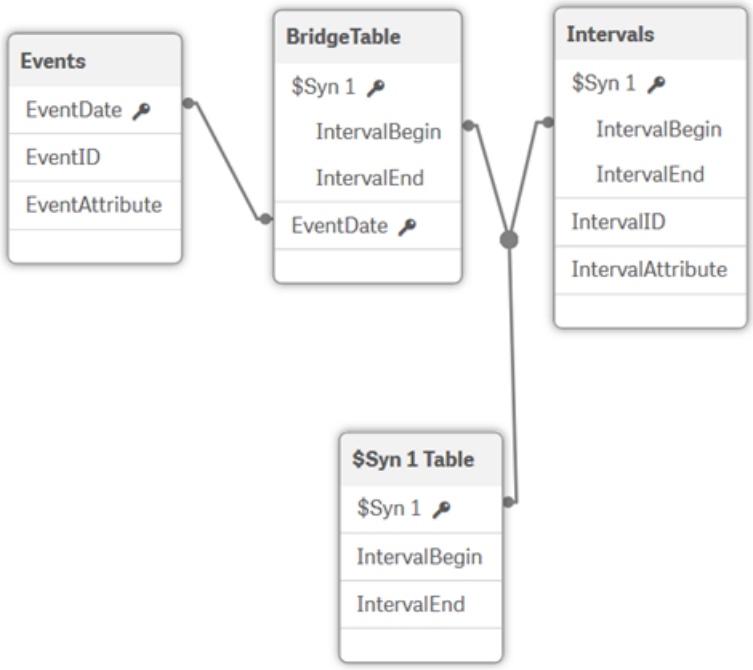
    ```
    BridgeTable: IntervalMatch (EventDate) LOAD distinct IntervalBegin, IntervalEnd Resident Intervals;
    ```

11. Скрипт должен выглядеть следующим образом:

    ```
    Events: LOAD     EventID,     EventDate,     EventAttribute FROM [lib://AttachedFiles/Events.txt]  (txt, utf8, embedded labels, delimiter is '\t', msq);
     Intervals: LOAD     IntervalID,     IntervalAttribute,     IntervalBegin, IntervalEnd FROM [lib://AttachedFiles/Intervals.txt]  (txt, utf8, embedded labels, delimiter is '\t', msq);   BridgeTable: IntervalMatch (EventDate) LOAD distinct IntervalBegin, IntervalEnd Resident Intervals;
    ```

12. Щелкните команду **Загрузить данные**.
13. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:

Модель данных: Таблицы Events, BridgeTable, Intervals и $Syn1



Модель данных содержит составной ключ (поля *IntervalBegin* и *IntervalEnd*), который станет синтетическим ключом Qlik Sense.

Основные таблицы:

- Таблица *Events* содержит только по одной записи для каждого события.
- Таблица *Intervals* содержит только по одной записи для каждого интервала.
- Таблица пересчета, которая содержит только по одной записи для комбинации события и интервала и объединяет две предыдущие таблицы.

Обратите внимание, что событие может принадлежать нескольким интервалам, если интервалы накладываются друг на друга. Также, разумеется, к интервалу могут относиться несколько событий.

Эта модель данных является оптимальной в том смысле, что она нормализована и компактна. Обе таблицы *Events* и *Intervals* остались неизменны и содержат исходное число записей. Все вычисления Qlik Sense, выполняемые в этих таблицах, например Count(EventID), будут работать правильно.

Чтобы узнать дополнительные сведения о функции IntervalMatch(), см. эту запись блога в Qlik Community: Using IntervalMatch() *(Использование IntervalMatch())*

Использование цикла While и итеративной загрузки IterNo()

Почти такую же таблицу пересчета можно получить с помощью цикла While с функцией IterNo(), который создает счетные значения между нижней и верхней границами интервала.

Цикл внутри оператора LOAD можно создать с помощью предложения While: Пример.

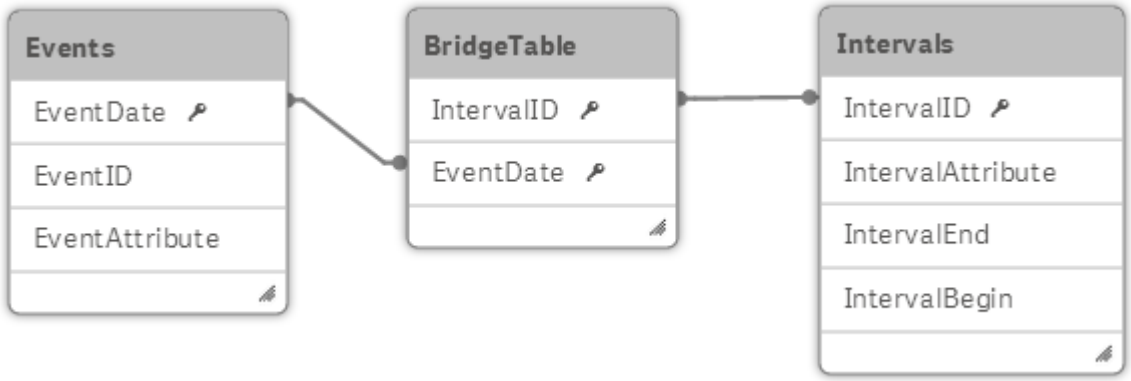
```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

Такой оператор LOAD будет охватывать каждую введенную запись и выполнять ее загрузку несколько раз до тех пор, пока выражение в предложении While является истинным. Функция IterNo() возвращает значение «1» в первой итерации, «2» во второй итерации и так далее.

У пользователя есть основной ключ для интервалов, IntervalID, поэтому единственным отличием этого скрипта будет способ создания таблицы пересчета:

Выполните следующие действия.

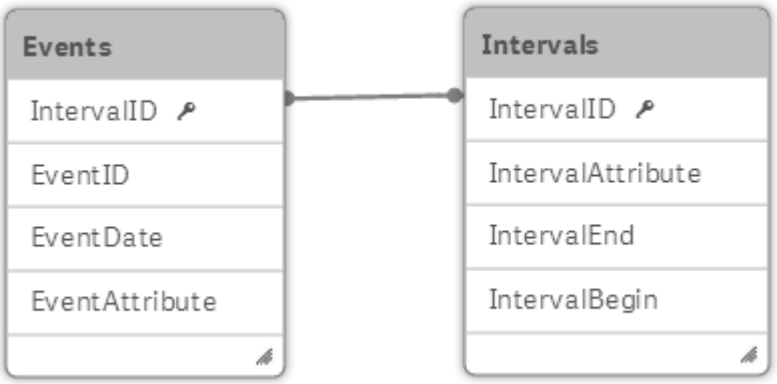
1. Замените существующие операторы Bridgetable следующим скриптом.
   ```
   BridgeTable: LOAD distinct * Where Exists(EventDate); LOAD IntervalBegin + IterNo() - 1
   as EventDate, IntervalID     Resident Intervals     While IntervalBegin + IterNo() - 1
   <= IntervalEnd;
   ```
2. Щелкните команду **Загрузить данные**.
3. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:
   Модель данных: Таблицы Events, BridgeTable и Intervals



   Обычно лучшим решением является использование трех таблиц, поскольку это позволяет создавать отношения между множествами, т. е. интервалами и событиями. Но часто возникает ситуация, когда пользователь знает, что событие может принадлежать только одному интервалу. В этом случае в таблице пересчета нет необходимости. *IntervalID* можно сохранить непосредственно в таблице событий. Существует несколько способов, но наиболее удобный — объединить таблицы Bridgetable и *Events*.

4. Добавьте следующий скрипт в конец вашего скрипта:
   ```
   Join (Events) LOAD EventDate, IntervalID Resident BridgeTable;   Drop Table BridgeTable;
   ```

5. Щелкните команду **Загрузить данные**.
6. Откройте раздел **Просмотр модели данных**. Модель данных выглядит так:
   Модель данных: Таблицы Events и Intervals



Открытые и закрытые интервалы

Тип интервала (закрытый/открытый) определяется конечными точками, а именно, включены они в интервал или нет.

- Если конечные точки включены в интервал, это закрытый интервал:
  $[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$
- Если конечные точки не включены в интервал, это открытый интервал:
  $]a,b[= \{x \in \mathbb{R} \mid a < x < b\}$
- Если одна конечная точка включена в интервал, это полуоткрытый интервал:
  $[a,b[= \{x \in \mathbb{R} \mid a \leq x < b\}$

Если интервалы накладываются друг на друга, а число может принадлежать нескольким интервалам, необходимо использовать закрытые интервалы.

Тем не менее, бывают ситуации, когда пользователю не нужны накладывающиеся интервалы, и число должно принадлежать только одному интервалу. Следовательно, если точка является одновременно концом одного интервала и началом другого, возникает проблема. Число с этим значением будет принадлежать обоим интервалам. Поэтому в таком случае необходимо использовать полуоткрытые интервалы.

Разумным решением данной проблемы является вычитание очень небольшой суммы из конечного значения всех интервалов, создавая, таким образом, закрытые, но не перекрывающиеся интервалы. Если используемые числа являются датами, самым простым решением будет использование функции DayEnd(), которая возвращает последнюю миллисекунду дня:

```
Intervals: LOAD…, DayEnd(IntervalEnd – 1) as IntervalEnd From Intervals;
```

Также можно вычесть небольшое количество вручную. При вычитании убедитесь, что вычитаемая сумма не слишком мала, поскольку операция будет округлена до 52 значимых двоичных значений (14 десятичных значений). Если использовать слишком малое количество, разница будет незначительной, поэтому снова будет использоваться исходное число.

4 Очистка данных

Бывают случаи, когда исходные данные, загруженные в Qlik Sense, имеют вид, отличный от того, как они должны выглядеть в приложении Qlik Sense. Qlik Sense предоставляет хост функций и операторов, которые позволяют преобразовывать данные в рабочий формат.

В скрипте Qlik Sense можно использовать сопоставление для замены или изменения значений полей или имен, когда запущен скрипт. Таким образом, сопоставление может использоваться для очистки и согласования данных либо для замены части или всех значений полей.

При загрузке данных из разных таблиц значения полей, обозначающие одно и то же, не всегда имеют одинаковые имена. Поскольку такая разнородность препятствует связыванию, подобную проблему необходимо решать. Это можно сделать достаточно просто, создав таблицу сопоставления для сравнения значений полей.

4.1 Таблицы сопоставления

Таблицы, загружаемые с помощью оператора Mapping load или Mapping select, обрабатываются отлично от других таблиц. Они сортируются в отдельной области памяти и используются только в качестве таблиц сопоставления в ходе выполнения скрипта. После выполнения скрипта эти таблицы автоматически исключаются.

Правила:

- Таблица сопоставления должна состоять из двух столбцов, первый из которых содержит значения, используемые для сравнения, а второй — желаемые значения для сопоставления.
- Двум столбцам следует присвоить имена, но имена сами по себе не важны. Имена столбцов не связаны с именами полей в обычных внутренних таблицах.

4.2 Функции и операторы Mapping

В данном учебном пособии описаны следующие операторы/функции сопоставления.

• Префикс Mapping

• ApplyMap()

• MapSubstring()

• Оператор Map … Using

• Оператор Unmap

4.3 Префикс Mapping

Префикс Mapping используется в скрипте для создания таблицы сопоставления. Затем таблицу сопоставления можно использовать с функцией ApplyMap(), MapSubstring() или оператором Map … Using.

Выполните следующие действия.

1. Создайте новое приложение и дайте ему имя.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите раздел *Countries*.
4. Введите следующий скрипт:
   ```
   CountryMap: MAPPING LOAD * INLINE [ Country, NewCountry U.S.A., US U.S., US United States, US United States of America, US ];
   ```

   Таблица *CountryMap* содержит два столбца: *Country* и *NewCountry*. Столбец *Country* содержит различные способы введения стран в поле *Country*. Столбец *NewCountry* хранит способ сопоставления этих значений. Эта таблица сопоставления будет использоваться для хранения согласованных значений страны *US* в поле *Country*. Например, если значение *U.S.A.* хранится в поле *Country*, сопоставьте его со значением *US*.

4.4 ApplyMap() функция

Используйте оператор ApplyMap(), чтобы заменить данные в поле, основываясь на ранее созданной таблице сопоставления. Таблицу сопоставления необходимо загрузить до использования функции ApplyMap(). Данные в таблице *Data.xlsx*, которые будут загружаться, выглядят следующим образом:

Таблица данных

ID	Имя	Страна	Код
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	United States	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

Обратите внимание, что название страны вводилось различными способами. Чтобы согласовать поля «country», загружается таблица сопоставления и используется функция **ApplyMap()**.

Выполните следующие действия.

1. Ниже введенного выше скрипта выберите и загрузите *Data.xlsx*, а затем вставьте данный скрипт.
2. Введите следующее над вновь созданным оператором LOAD:
   ```
   Data:
   ```

   Скрипт должен выглядеть следующим образом:

   ```
   CountryMap: MAPPING LOAD * INLINE [     Country, NewCountry     U.S.A., US     U.S., US
      United States, US     United States of America, US ];         Data: LOAD     ID,     Name,
      Country,     Code FROM [lib://AttachedFiles/Data.xlsx] (ooxml, embedded labels, table
   ```

```
is Sheet1);
```

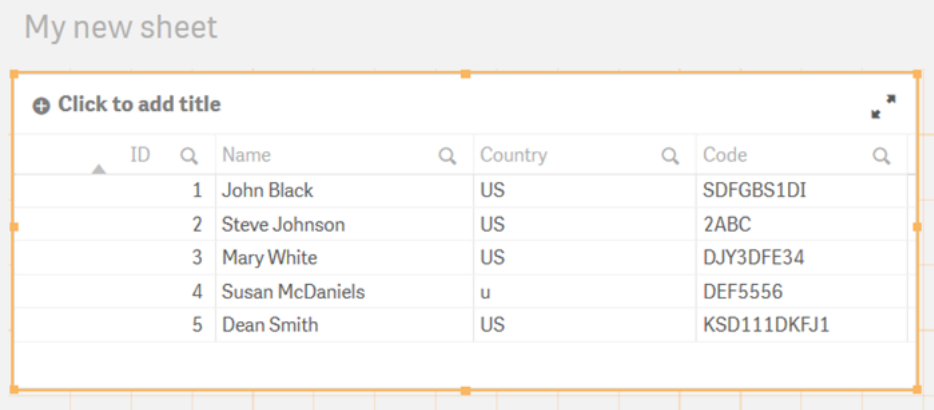
3. Измените строку, содержащую `Country,`, следующим образом:
   `ApplyMap('CountryMap', Country) as Country,`

   Первый параметр функции ApplyMap() — это имя сопоставления, заключенное в одинарные кавычки. Второй параметр — это поле с данными, которые необходимо заменить.

4. Щелкните команду **Загрузить данные**.

   Полученная таблица выглядит следующим образом:

   Таблица, в которой отображаются данные, загруженные с помощью функции ApplyMap()

ID	Name	Country	Code
1	John Black	US	SDFGBS1DI
2	Steve Johnson	US	2ABC
3	Mary White	US	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

   Все различные варианты написания *United States* были заменены на *US*. Есть одна запись, которая была написана неправильно, поэтому функция ApplyMap() не изменила значение этого поля. В функции ApplyMap() можно использовать третий параметр для добавления выражения по умолчанию, если в таблице сопоставления нет соответствующего значения.

5. Добавьте `'US'` в качестве третьего параметра функции ApplyMap(), чтобы обрабатывать такие случаи, когда страна указана неправильно.
   `ApplyMap('CountryMap', Country, 'US') as Country,`

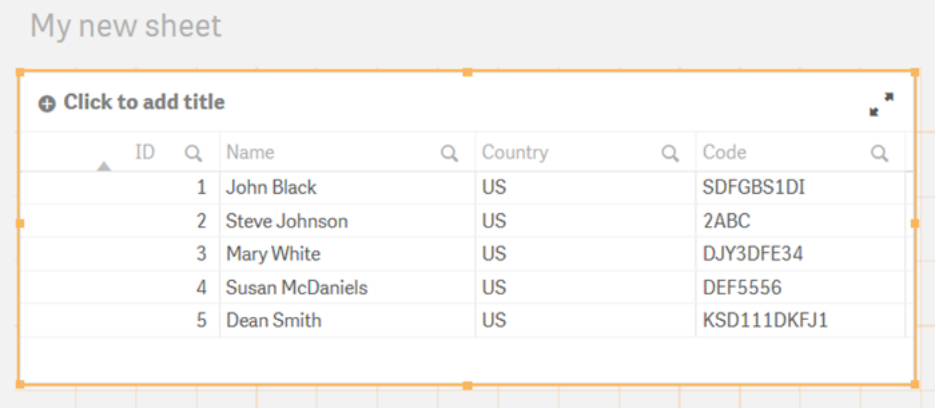
   Скрипт должен выглядеть следующим образом:

```
CountryMap: MAPPING LOAD * INLINE [     Country, NewCountry     U.S.A., US     U.S., US
   United States, US     United States of America, US ];  Data: LOAD     ID,     Name,
  ApplyMap('CountryMap', Country, 'US') as Country,     Code FROM
[lib://AttachedFiles/Data.xlsx] (ooxml, embedded labels, table is Sheet1);
```

6. Щелкните команду **Загрузить данные**.

   Полученная таблица выглядит следующим образом:

Таблица, в которой отображаются данные, загруженные с помощью функции ApplyMap



Чтобы узнать дополнительные сведения о функции ApplyMap(), см. эту запись блога в Qlik Community: Не объединяйте — используйте вместо этого функцию Applymap

4.5 MapSubstring() функция

Функция MapSubstring() позволяет сопоставлять части поля.

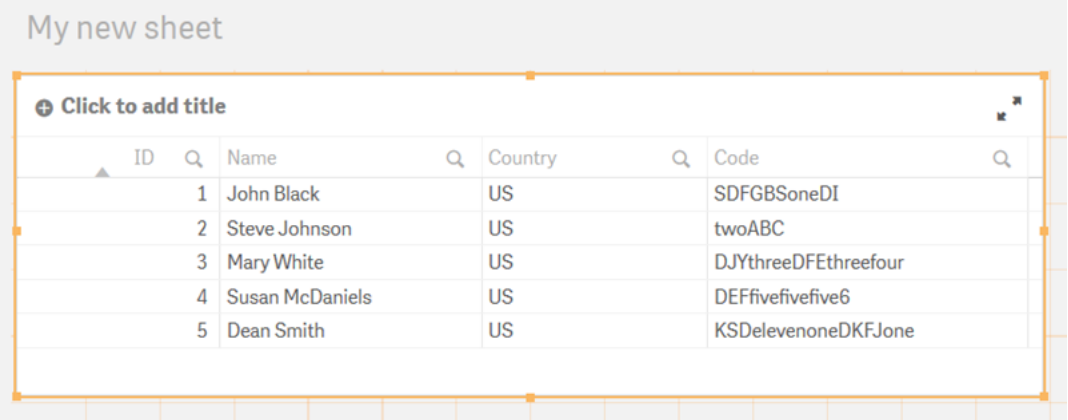
В таблице, созданной функцией ApplyMap(), нам теперь необходимо, чтобы числа записывались в виде текста. Таким образом, функция MapSubstring() будет использоваться для замены числовых данных тестом.

Для этого сначала необходимо создать таблицу сопоставления.

Выполните следующие действия.

1. Добавьте следующие строки скрипта после раздела *CountryMap*, но перед разделом *Data*.
   ```
   CodeMap: MAPPING LOAD * INLINE [ F1, F2 1, one 2, two 3, three 4, four 5, five 11, eleven ];
   ```
   В таблице *CodeMap* сопоставлены числа от 1 до 5, и 11.
2. В разделе скрипта *Data* измените оператор Code следующим образом:
   ```
   MapSubString('CodeMap', Code) as Code
   ```
   Скрипт должен выглядеть следующим образом:
   ```
   CountryMap: MAPPING LOAD * INLINE [     Country, NewCountry     U.S.A., US     U.S., US     United States, US     United States of America, US ];  CodeMap: MAPPING LOAD * INLINE [ F1, F2 1, one 2, two 3, three 4, four 5, five 11, eleven ];  Data: LOAD     ID,     Name,     ApplyMap('CountryMap', Country, 'US') as Country,     MapSubString('CodeMap', Code) as Code FROM [lib://AttachedFiles/Data.xlsx] (ooxml, embedded labels, table is Sheet1);
   ```
3. Щелкните команду **Загрузить данные**.
   Полученная таблица выглядит следующим образом:

Таблица, в которой отображаются данные, загруженные с помощью функции MapSubString



В поле *Code* числовые символы были заменены текстом. Если число повторяется несколько раз, как для ID=3 и ID=4, текст также повторяется. ID=4. *Susan McDaniels* имеет в коде цифру 6. Поскольку цифра 6 не была сопоставлена в таблице *CodeMap*, она остается без изменений. ID=5, *Dean Smith* имеет в коде 111. Этому числу сопоставлен текст 'elevenone'.

Чтобы узнать дополнительные сведения о функции MapSubstring(), см. эту запись блога в Qlik Community: Сопоставление … не географического типа

4.6 Map … Using

Оператор Map … Using может быть также использован для применения сопоставления к полю. Однако он работает немного по-другому, чем функция ApplyMap(). В то время как ApplyMap() выполняется при каждом появлении имени поля в выражении, Map … Using применяет сопоставление, если значение сохранено во внутренней таблице под определенным именем поля.

Давайте разберем пример. Допустим, что мы несколько раз загружали поле *Country* в скрипт и хотели, чтобы сопоставление применялось при каждой загрузке поля. Можно использовать функцию ApplyMap (), как рассматривалось ранее в данном учебном пособии, или можно использовать Map … Using.

Если Map … Using используется, когда сопоставление применяется к полю, это поле сохраняется во внутреннюю таблицу. Таким образом, в примере ниже сопоставление применяется к полю *Country* в таблице Data1, но не применяется к полю *Country2* в таблице *Data2*. Это происходит, потому что оператор Map … Using применяется только к полям с именем *Country*. Когда поле *Country2* сохраняется во внутреннюю таблицу, его имя уже не *Country*. Чтобы сопоставление применялось к таблице *Country2*, необходимо использовать функцию ApplyMap().

Оператор Unmap завершает работу оператора Map … Using, поэтому, если поля *Country* были загружены после оператора Unmap, функция *CountryMap* не будет применена.

Выполните следующие действия.

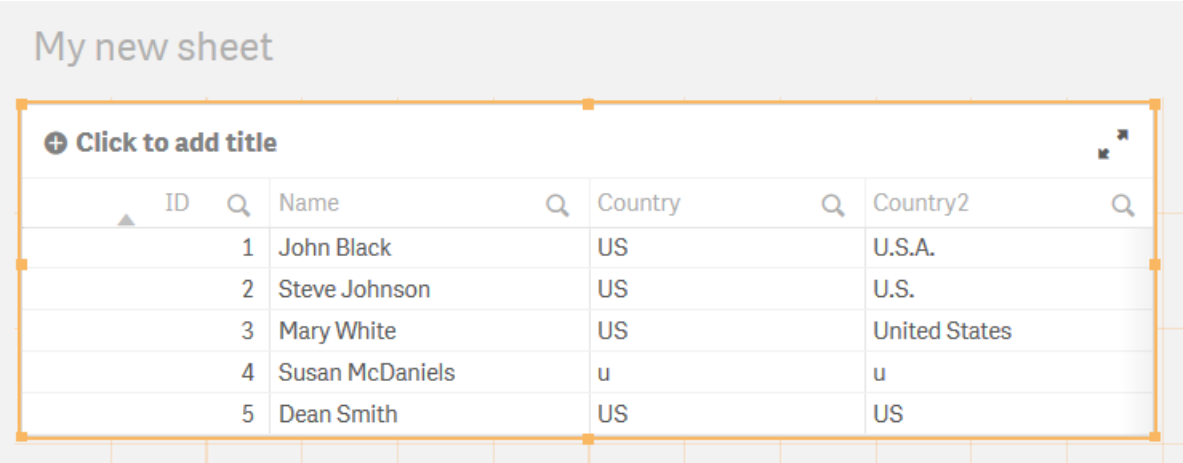
1. Замените скрипт для таблицы *Data* на следующее:

   ```
   Map Country Using CountryMap; Data1:     LOAD     ID,     Name,     Country FROM
   [lib://AttachedFiles/Data.xlsx] (ooxml, embedded labels, table is Sheet1);
   LOAD     ID,     Country as Country2 FROM [lib://AttachedFiles/Data.xlsx] (ooxml,
   embedded labels, table is Sheet1); UNMAP;
   ```

2. Щелкните команду **Загрузить данные**.

   Полученная таблица выглядит следующим образом:

   Таблица, в которой отображаются данные, загруженные с помощью функции Map … Using



5 Обработка иерархических данных

Иерархии — это важная часть всех решений бизнес-анализа, используемых для описания измерений, которые, разумеется, содержат различные уровни модульности. Одни из них просты и интуитивно понятны, тогда как другие сложны, и их моделирование требует значительных усилий.

От верхней до нижней части иерархии члены иерархии становятся более детализированными. Например, в измерении с уровнями «Рынок», «Страна», «Штат» или «Город» член «Америка» появляется на верхнем уровне иерархии, член «США» появляется на втором уровне, член «Калифорния» — на третьем, а «Сан-Франциско» — на нижнем уровне. «Калифорния» — более конкретное понятие, чем «США», а «Сан-Франциско» более конкретное, чем «Калифорния».

Сохранение иерархий в реляционной модели — это обычная задача со множеством решений. Существует несколько подходов:

- Горизонтальная иерархия
- Модель списка смежных вершин
- Способ перечисления путей
- Модель вложенных наборов
- Родительский список

В данном учебном пособии мы будем создавать родительский список, поскольку иерархия в нем представлена в том виде, который как раз используется в запросах. Дополнительную информацию о других подходах к решению данной задачи см. в разделе Qlik Community.

5.1 Префикс Hierarchy

Префикс Hierarchy является командой скрипта, которая ставится перед оператором LOAD или SELECT, который загружает таблицу со смежными узлами. У оператора LOAD должно быть как минимум три поля: идентификатор, который является уникальным ключом к узлу, ссылка на родителя и имя.

С помощью префикса загруженная таблица будет преобразована в таблицу расширенных узлов; таблицу с дополнительными столбцами, по одному для каждого уровня иерархии.

Выполните следующие действия.

1. Создайте новое приложение и дайте ему имя.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Вызовите раздел *Wine*.
4. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
5. Загрузите, а затем выберите *Winedistricts.txt*.
6. В окне **Выбор данных из** снимите флажки для полей *Lbound* и *RBound*, чтобы не осуществлялась их загрузка.
7. Щелкните команду **Вставить скрипт**.

8. Введите следующее над оператором LOAD:
   ```
   Hierarchy (NodeID, ParentID, NodeName)
   ```

   Скрипт должен выглядеть следующим образом:

   ```
   Hierarchy (NodeID, ParentID, NodeName) LOAD     NodeID,     ParentID,     NodeName FROM [lib://AttachedFiles/Winedistricts.txt]  (txt, utf8, embedded labels, delimiter is '\t', msq);
   ```

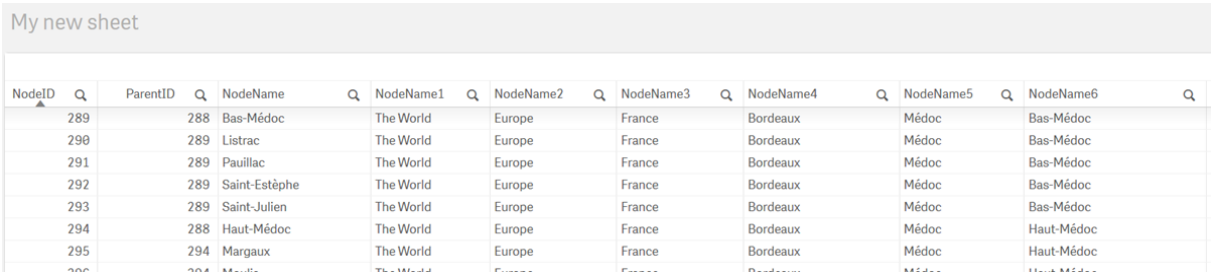
9. Щелкните команду **Загрузить данные**.
10. Используйте раздел **Предварительный просмотр просмотра модели данных**, чтобы просмотреть полученную таблицу.

    Полученная таблица расширенных узлов имеет то же число записей, что и исходная таблица: по одной записи на узел. Таблица расширенных узлов очень практична, поскольку она отвечает ряду следующих требований для анализа иерархии в реляционной модели.

    - Все имена узлов находятся в одном и том же столбце, поэтому это можно использовать для поиска.
    - Кроме того, различные уровни узлов были расширены так, что у каждого из них по одному полю; эти поля можно использовать в группах детализации или в качестве измерений в сводных таблицах.
    - Кроме того, различные уровни узлов были расширены так, что у каждого из них по одному полю; эти поля можно использовать в группах детализации.
    - Это можно сделать, чтобы указать путь, уникальный для каждого узла, с перечислением всех родителей в правильном порядке.
    - Это можно сделать, чтобы указать глубину узла, т. е. расстояние от корневой папки.

    Полученная таблица выглядит следующим образом:

    Таблица, в которой отображаются данные образца, загруженные с помощью префикса Hierarchy

    
    My new sheet

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	NodeName4	NodeName5	NodeName6
289	288	Bas-Médoc	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
290	289	Listrac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
291	289	Pauillac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
292	289	Saint-Estèphe	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
293	289	Saint-Julien	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
294	288	Haut-Médoc	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc
295	294	Margaux	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc

Чтобы узнать дополнительные сведения об иерархиях, см. эту запись блога в Qlik Community: Иерархии

5.2 Префикс HierarchyBelongsTo

Как и префикс Hierarchy, префикс HierarchyBelongsTo представляет собой команду скрипта, которая помещается перед оператором LOAD или SELECT, загружающим таблицу со смежными узлами.

Также у оператора LOAD должно быть как минимум три поля: идентификатор, который является уникальным ключом к узлу, ссылка на родителя и имя. С помощью префикса загруженная таблица будет преобразована в родительскую таблицу, таблицу со всевозможными комбинациями родителя и потомка, указанными в виде отдельной записи. Таким образом, очень просто найти всех родителей или всех потомков определенного узла.

Выполните следующие действия.

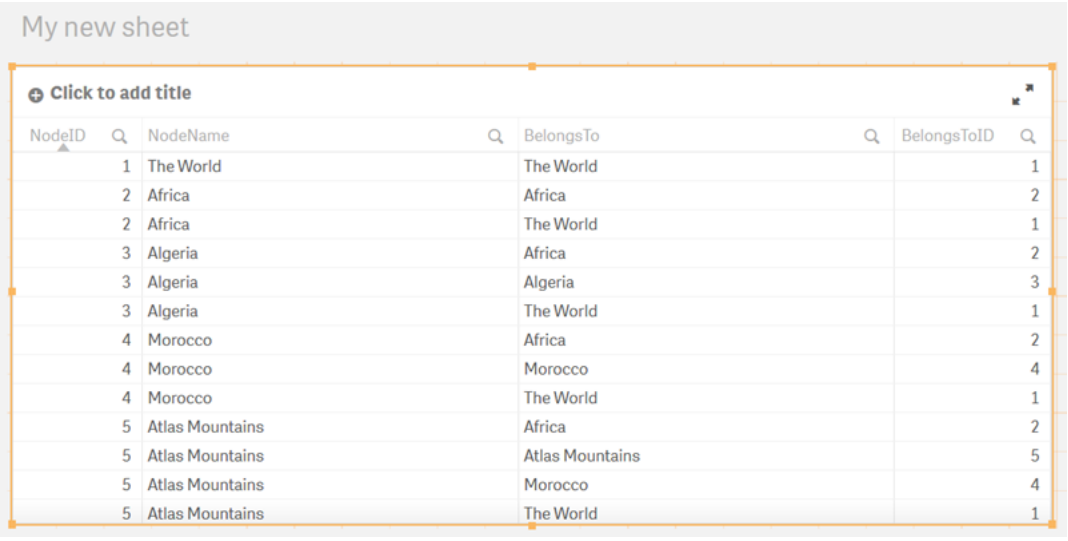
1. Измените оператор Hierarchy в **редакторе загрузки данных**, чтобы он выглядел так:
   `HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)`
2. Щелкните команду **Загрузить данные**.
3. Используйте раздел **Предварительный просмотр просмотра модели данных**, чтобы просмотреть полученную таблицу.

   Родительская таблица отвечает ряду требований для анализа иерархии в реляционной модели:

   - Если идентификатор узла представляет одиночные узлы, то идентификатор родителя представляет все дерево и поддеревья иерархии.
   - Все имена узлов относятся одновременно к узлам и деревьям, и их можно использовать для поиска в обеих этих ролях.
   - Это можно сделать, чтобы указать глубину разницы между глубиной узла и глубиной родителя, т. е. расстояние от корневой папки до поддерева.

   Полученная таблица выглядит следующим образом:

   Таблица, в которой отображаются данные, загруженные с помощью префикса HierarchyBelongsTo


My new sheet

Click to add title

NodeID	NodeName	BelongsTo	BelongsToID
1	The World	The World	1
2	Africa	Africa	2
2	Africa	The World	1
3	Algeria	Africa	2
3	Algeria	Algeria	3
3	Algeria	The World	1
4	Morocco	Africa	2
4	Morocco	Morocco	4
4	Morocco	The World	1
5	Atlas Mountains	Africa	2
5	Atlas Mountains	Atlas Mountains	5
5	Atlas Mountains	Morocco	4
5	Atlas Mountains	The World	1

Авторизация

Довольно часто иерархия используется для авторизации. Одним из примеров является иерархия организационной структуры. Каждый менеджер имеет право видеть всю информацию, которая относится к его отделу, включая подчиненные отделы. Но менеджерам не обязательно видеть информацию, которая касается других отделов.

Пример иерархии организационной структуры

Это значит, что разные люди будут иметь доступ к разным поддеревьям организации. Таблица авторизации может выглядеть так:

Таблица авторизации

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
USER	ACME\JRL	John	CPO	HR
USER	ACME\CAH	КэролCarol	CEO	CEO
USER	ACME\JER	James	Director Engineering	Engineering
USER	ACME\DBK	Diana	CFO	Finance
USER	ACME\RNL	Bob	COO	Sales
USER	ACME\LFD	Larry	CTO	Продукт

В этом случае *Carol* может видеть все, что относится к *CEO* и ниже; *Larry* может видеть организацию *Product*; а *James* может видеть только организацию *Engineering*.

Пример:

Нередко для хранения иерархии используется таблица смежных узлов. В этом примере для решения этой задачи можно загрузить таблицу смежных узлов при помощи HierarchyBelongsTo и присвоить родительскому полю имя *Tree*.

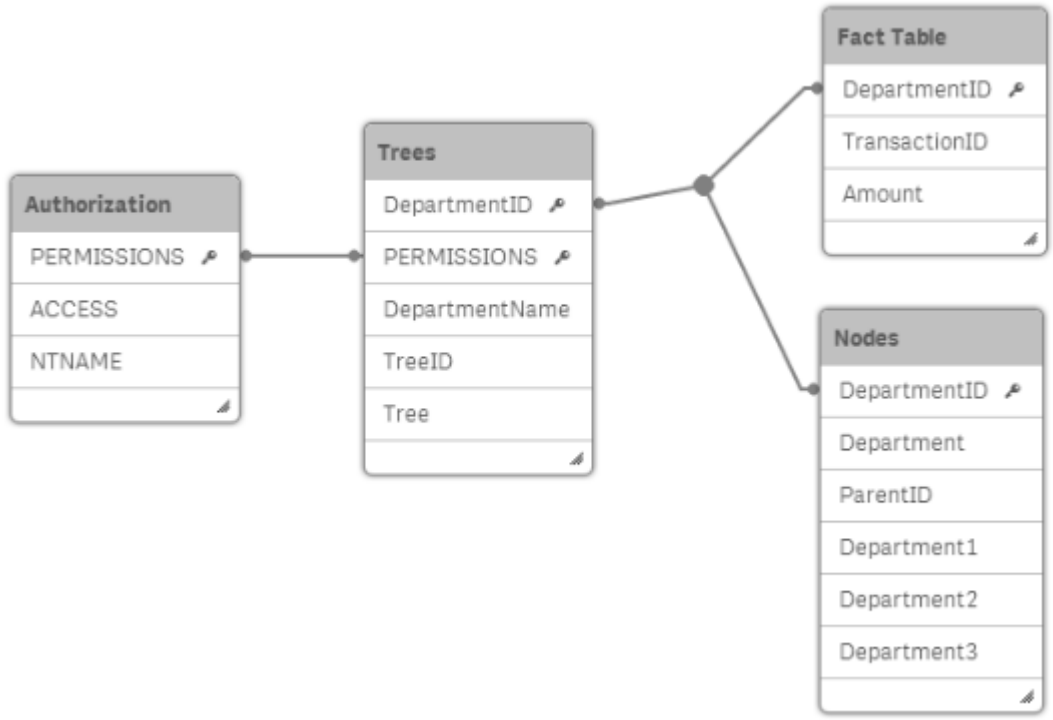
Для использования Section Access загрузите копию поля *Tree* и присвойте этому новому полю имя *PERMISSIONS*. И наконец, следует загрузить таблицу авторизации. Выполнить два последних шага можно с помощью следующих строк скрипта. Обратите внимание, что таблица TempTrees является таблицей, созданной оператором HierarchyBelongsTo.

Обратите внимание, что это просто пример. Здесь нет сопутствующих упражнений, которые следует выполнить в Qlik Sense.

```
Trees: LOAD *,     Upper(Tree) as PERMISSIONS     Resident TempTrees; Drop Table TempTrees;
Section Access; Authorization: LOAD     ACCESS,         NTNAME,         UPPER(Permissions) as PERMISSIONS Fro
Organization; Section Application;
```

В этом примере будет получена следующая модель данных:

Модель данных: Таблицы Authorization, Trees, Fact и Nodes



6 Файлы QVD

Файл QVD (QlikView Data) — это файл, в котором содержится таблица данных, экспортируемых из программы Qlik Sense или QlikView. QVD является собственным форматом Qlik и может быть записан и прочтен только с помощью Qlik Sense или QlikView. Формат файла оптимизирован для скорости при чтении данных из скрипта Qlik Sense, но в то же время не занимает много места. Чтение данных из файла QVD обычно в 10–100 раз быстрее, чем чтение из других источников данных.

Файлы QVD можно читать в двух режимах: стандартном (быстром) и оптимизированном (сверхбыстром). Выбор режима выполняется обработчиком скриптов Qlik Sense автоматически. Оптимизированный режим может использоваться только в том случае, если все загруженные поля считываются без преобразований (формул, действующих на поля), но в то же время допускается изменение имен полей. Предложение Where в программе Qlik Sense управляет распаковкой записей и отключает оптимизированную загрузку.

Файл QVD содержит только одну таблицу данных и состоит из трех частей:

- Заголовок XML (в кодировке UTF-8), который описывает поля в таблице, макета последующей информации и некоторых прочих метаданных.
- Таблицы символов в формате, заполненном байтами.
- Фактические данные таблиц в формате, заполненном битами.

Файлы QVD используются для многих целей. Можно выделить четыре главных области назначения. В любой из этих ситуаций может использоваться несколько из них:

- Увеличение скорости загрузки данных

  Для больших наборов данных выполнение скрипта значительно ускоряется благодаря буферизации неизменяющихся или медленно изменяющихся наборов входных данных в файлах QVD.

- Увеличение скорости загрузки данных

  Для больших наборов данных выполнение скрипта значительно ускоряется благодаря буферизации неизменяющихся или медленно изменяющихся наборов входных данных в файлах QVD.

- Снижение нагрузки на серверы баз данных

  Объем данных, выбираемых из внешних источников данных, может также значительно сократиться. Это сокращает рабочую нагрузку на внешние базы данных и сетевой трафик. Более того, если несколько скриптов Qlik Sense совместно используют одни и те же данные, необходимо лишь один раз загрузить их из исходной базы данных в файл QVD. Другие приложения могут использовать те же данные с помощью файла QVD.

- Консолидирование данных из нескольких приложений Qlik Sense.

При работе с оператором скрипта Binary можно загрузить данные только из одного приложения Qlik Sense в другое, но с файлами QVD скрипт Qlik Sense может совместно использовать данные из любого числа приложений Qlik Sense. Это открывает возможности для приложений по консолидации похожих данных из разных подразделений компании и т. д.

- Инкрементальная загрузка

  Часто функции QVD могут использоваться для облегчения инкрементальной загрузки, то есть загрузки только новых записей из постоянно растущей базы данных.

6.1 Создание файлов QVD

Файл QVD можно создать двумя способами:

- Явное создание и присвоение имен с помощью команды Store в скрипте Qlik Sense.

  Укажите в скрипте, что ранее считанную таблицу или ее часть необходимо экспортировать в явно названный файл в указанном вами местоположении.

- Автоматическое создание и обслуживание из скрипта.

  Путем постановки перед оператором load или select префикса Buffer приложение Qlik Sense автоматически создает файл QVD, который в некоторых случаях может использоваться вместо оригинального источника данных при перезагрузке данных.

Между итоговыми файлами QVD нет различий относительно скорости чтения.

Store

Эта функция скрипта создает файл QVD, CSV или txt с заданным именем.

Синтаксис:

```
Store[ *fieldlist from] table into filename [ format-spec ];
```

Оператор может экспортировать поля только из одной таблицы данных. Если требуется экспортировать поля из нескольких таблиц, необходимо заранее сформировать явное объединение в скрипте для создания таблицы данных, которую следует экспортировать.

Текстовые значения экспортируются в файл CSV в формате UTF-8. Можно указать разделитель. См. **LOAD**. Оператор store для файла CSV не поддерживает экспорт BIFF .

Примеры:

```
Store mytable into [lib://AttachedFiles/xyz.qvd];
Store * from mytable into [lib://FolderConnection/xyz.qvd];
Store myfield from mytable into 'lib://FolderConnection/xyz.qvd';
Store myfield as renamedfield, myfield2 as renamedfield2 from mytable into
[lib://AttachedFiles/xyz.qvd];
Store mytable into 'lib://FolderConnection/myfile.txt';
Store * from mytable into 'lib://FolderConnection/myfile.csv';
```

Выполните следующие действия.

1. Откройте приложение *Учебное пособие по написанию скриптов продвинутого уровня*.
2. Щелкните раздел скрипта *Product*.
3. Добавьте следующее в конец данного скрипта:
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

   Скрипт должен выглядеть следующим образом:

   ```
   CrossTable(Month, Sales)
   LOAD
       Product,
       "Jan 2014",
       "Feb 2014",
       "Mar 2014",
       "Apr 2014",
       "May 2014"
   FROM [lib://AttachedFiles/Product.xlsx]
   (ooxml, embedded labels, table is Product);

   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

4. Щелкните команду **Загрузить данные**.
   В списке файлов должен появиться файл *Product.qvd*.

   Этот файл данных является результатом скрипта **Crosstable** и представляет собой таблицу из трех столбцов, по одному столбцу для каждой категории (Product, Month, Sales). Этот файл данных теперь можно использовать для замены целого раздела скрипта *Product* .

6.2 Чтение данных из файлов QVD

Программа Qlik Sense может читать файл QVD или получать к нему доступ следующими способами:

- Загрузка файла QVD в качестве явного источника данных. Оператор load может ссылаться на файлы QVD в скрипте Qlik Sense, как на любые другие типы текстовых файлов (csv, fix, dif, biff и т. д.).

  Примеры:

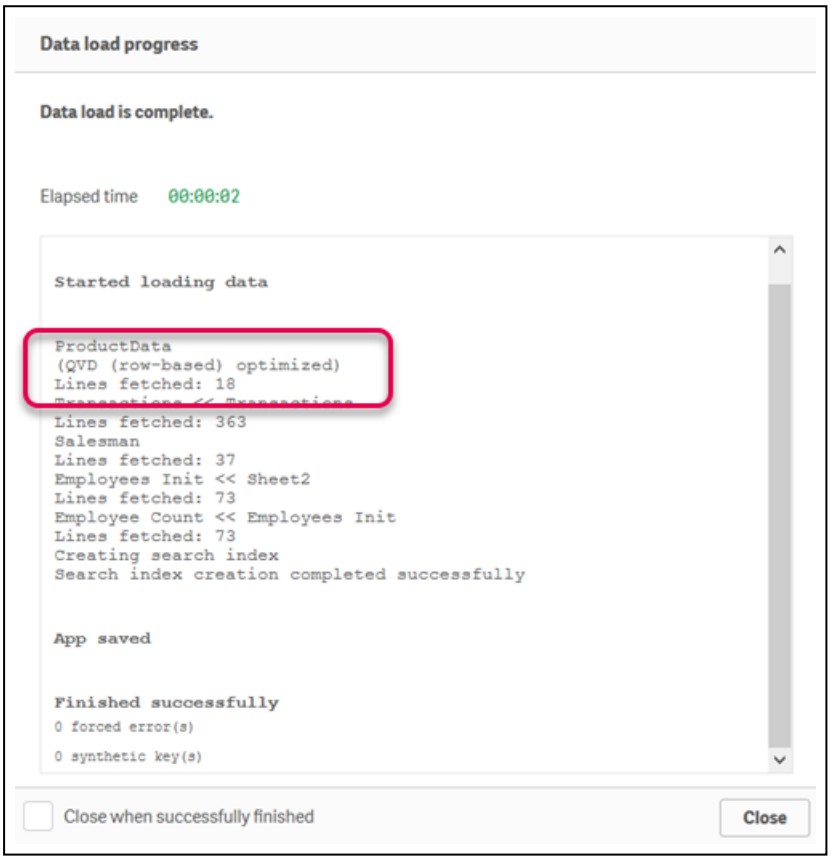
  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

- Автоматическая загрузка буферизованных файлов QVD. При использовании префикса buffer с операторами load или select для чтения явные выражения не требуются. Программа Qlik Sense определяет степень, до которой она использует данные из файла QVD, в отличие от получения данных с помощью оригинального оператора LOAD или SELECT.
- Доступ к файлам QVD с помощью скрипта. Для получения различной информации о данных, находящихся в заголовке XML файла QVD может использоваться несколько функций скриптов (начинаются с QVD).

Выполните следующие действия.

1. Закомментируйте весь скрипт в разделе скрипта *Product*.
2. Введите следующий скрипт:
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
3. Щелкните команду **Загрузить данные**.
   Данные загружаются из файла QVD.

   Окно хода выполнения загрузки данных



Чтобы узнать об использовании файлов QVD для инкрементальных загрузок, см. эту запись блога в Qlik Community: Overview of Qlik Incremental Loading *(Обзор инкрементальной загрузки в Qlik)*

Buffer

Файлы QVD могут создаваться и обслуживаться автоматически посредством префикса Buffer. Этот префикс может использоваться на большинстве операторов LOAD и SELECT в скрипте. Он указывает на то, что файлы QVD используются для кэширования/буферизации результата оператора.

Синтаксис:

```
Buffer [ (option [ , option])] ( loadstatement | selectstatement )
       option::= incremental | stale [after] amount [(days | hours)]
```

Если не используется ни один параметр, буфер QVD, созданный при первом выполнении скрипта, будет использоваться в течение неопределенного времени.

Пример:

```
Buffer load * from MyTable;
```

stale [after] amount [(days | hours)]

Amount — число, обозначающее период времени. Могут использоваться десятичные числа Decimals. Единицей измерения являются дни, если не указано.

Параметр stale after обычно используется с источниками баз данных, где нет простой метки времени на оригинальных данных. Предложение stale after просто указывает период времени с момента создания буфера QVD, после которого оно будет считаться недействительным. До этого времени в качестве источника данных будет использоваться буфер QVD, а после этого — оригинальный источник данных. Файл буфера QVD будет автоматически обновлен, и начнется новый период.

Пример:

```
Buffer (stale after 7 days) load * from MyTable;
```

Incremental

Параметр incremental дает возможность прочитать только часть базового файла. Данные о предыдущем размере файла находятся в заголовке XML файла QVD. Это особенно полезно при работе с файлами журнала. Все записи, загруженные в предыдущий раз, считываются из файла QVD, в то время как последующие новые записи считываются из оригинального источника, в результате чего создается обновленный файл QVD.

Обратите внимание, что параметр incremental может использоваться только с операторами LOAD и текстовыми файлами, а инкрементальная загрузка не может использоваться там, где устаревшие данные изменены или удалены.

Пример:

```
Buffer (incremental) load * from MyLog.log;
```

Обычно буферы QVD удаляются, если к ним больше не обращаются ни на каком этапе выполнения всего скрипта в приложении, его создавшем, либо в том случае, если приложение, его создавшее, уже не существует. Оператор Store используется, если необходимо сохранить содержимое буфера в качестве файлов QVD или CSV.

Выполните следующие действия.

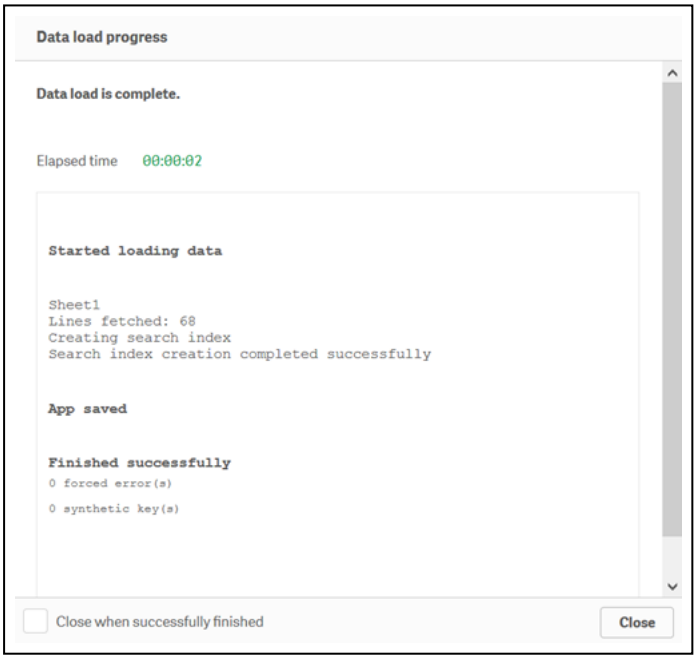
1. Создайте новое приложение и дайте ему имя.
2. Добавьте новый раздел скрипта в **Редакторе загрузки данных**.
3. Под элементом **AttachedFiles** в меню справа щелкните **Выбрать данные**.
4. Загрузите, а затем выберите *Cutlery.xlsx*.
5. В окне **Выбрать данные из** щелкните команду **Вставить скрипт**.
6. Закомментируйте поля в операторе LOAD и измените его следующим образом:
   `Buffer LOAD *`

   Скрипт должен выглядеть следующим образом:

   ```
   Buffer LOAD *
          //     "date",
          //     item,
          //     quantity
          FROM [lib://AttachedFiles/Cutlery.xlsx]
          (ooxml, embedded labels, table is Sheet1);
   ```

7. Щелкните команду **Загрузить данные**.
   При первой загрузке данных они загружаются из *Cutlery.xlsx*.

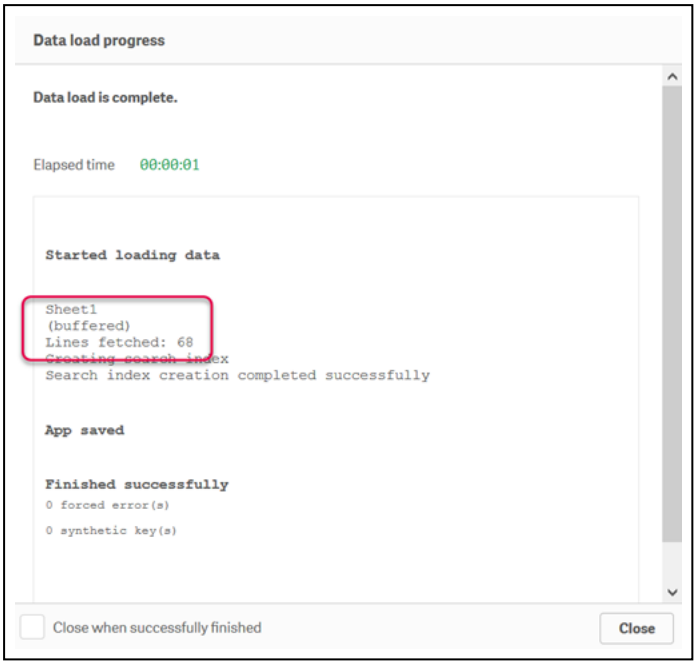
   Окно хода выполнения загрузки данных

   Оператор Buffer также создает файл QVD и сохраняет его в Qlik Sense. В развертывании Qlik Sense Enterprise on Windows он сохраняется в каталог на сервере Qlik Sense.

8. Щелкните еще раз **Загрузить данные**.
9. В этот раз данные загружаются из файла QVD, созданного оператором Buffer при первой загрузке данных.

Окно хода выполнения загрузки данных



6.3 Спасибо!

Вы закончили обучение по данному учебному пособию. Надеемся, вы узнали много нового о написании скриптов в программе Qlik Sense. Чтобы получить информацию о дальнейших курсах обучения, посетите наш веб-сайт.